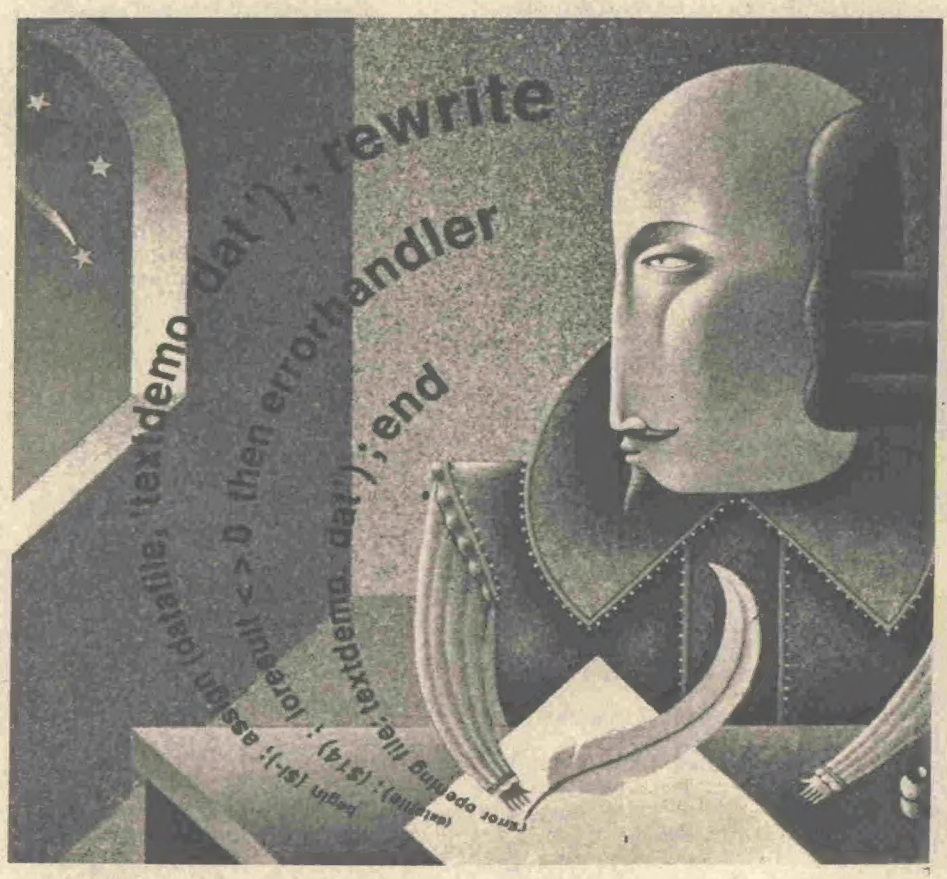
SOFT MAIL

สร้างคำสั่งฮอตไลน์ อ่าน/เขียนดิสค์ ให้เทอร์โบปาสคาล

ขจัดปัญหาหนักใจ ให้เทอร์โบปาสคาล ในการอ่าน/เขียนดิสค์ ด้วยคำสั่งสายตรง



ผู้ใช้เทอร์โบปาสคาลเป็นภาษาหากินคงจะเคยนึกน้อยใจที่เทอร์โบปาสคาลไม่มีคำสั่งในการอ่านและเขียนดิสค์โดยตรงเหมือนเทอร์โบซี บางท่านอาจคิดที่จะเขียนโปรแกรมวัคซีนตรวจเช็คไวรัสสำหรับใช้งานเอง แต่ติดปัญหาตรงที่ว่าไม่สามารถอ่านบูตเซกเตอร์มาตรวจสอบได้ หากจะหันไปเล่นเทอร์โบซีก็ต้องเสียเวลาอีกนานพอสมควรกว่าจะชำนาญเท่ากับเทอร์โบปาสคาล เราลองช่วยกันแก้ไขที่สาเหตุดีกว่าครับ

แนวทางแก้ปัญหาของเรามีให้เลือก 2 วิธี วิธีแรกคือการใช้คำสั่ง Intr ( ) ของเทอร์โบปาสคาลให้สร้างโพรซีเดอร์ใหม่ ส่วนอีกวิธีหนึ่งนั้นเราจะใช้ภาษาแอสเซมบลีโดยนำรูทีนภาษาแอสเซมบลีที่สร้างขึ้นมาอินเตอร์เฟสกับเทอร์โบปาสคาล ซึ่งความจริงแล้วทั้งสองวิธีนี้ก็คือการเรียกอินเตอร์รัพต์ของดอสมาใช้งานนั่นเอง แต่การทำเป็นรูทีนภาษาแอสเซมบลีจะทำให้โปรแกรมมีขนาดเล็กและรันได้เร็วกว่า ต่อไปเราจะมาดูกันว่าอินเตอร์รัพต์ใดบ้างที่เราจะนำมาใช้

อินเตอร์รัพต์ของดอสที่เราเรียกมาใช้งานก็คืออินเตอร์รัพต์ 25H ซึ่งอ่านจากดิสค์โดยตรง (Absolute Disk Read) และอินเตอร์รัพต์ 26H สำหรับเขียนลงบนดิสค์ ซึ่งก่อนที่จะเรียกใช้อินเตอร์รัพต์ทั้งสองนี้จะต้องตั้งค่าในรีจิสเตอร์ดังต่อไปนี้

AL หมายเลขไดรฟ์ที่จะอ่านหรือเขียน (0 = A, 1 = B)

CX จำนวนเซกเตอร์ที่จะอ่านเข้ามาในหน่วยความจำหรือเขียนลงบนดิสค์

DX หมายเลขเริ่มต้นของลอจิคัลเซกเตอร์

DS:BX ตำแหน่งที่อยู่เริ่มต้น (เซกเมนต์ : ออฟเซต) ของข้อมูลในหน่วยความจำ

ทีนี้เรามาเริ่มลงมือกันเลยดีกว่า

สร้างยูนิต SECTOR__ ไว้ใช้งาน

เริ่มจากการสร้างไฟล์ RDSECTOR.ASM และ WRSECTOR.ASM ดังรูปที่ 1 และรูปที่ 2 ขึ้นมาเสียก่อน แล้วนำไปคอมพายล์โดยแอสเซมเบลอร์ให้ได้ออบเจ็กต์ไฟล์ RDSECTOR.OBJ และ WRSECTOR.OBJ เสร็จแล้วสร้างไฟล์ SECTOR__.PAS ตามรูปที่ 3 หลังจากนั้นจึงเข้าไปในเทอร์โบปาสคาลเพื่อคอมพายล์ SECTOR__.PAS จะมีการลิงค์ออบเจ็กต์ไฟล์ทั้งสองเข้าด้วยกัน

ยูนิต SECTOR__ ที่สร้างขึ้นจะมี 2 โพรซีเดอร์ให้เราใช้งานคือ โพรซีเดอร์ ReadSector (Segment, Offset, Drive, Sector, Number : word); และ WriteSector (Segment, Offset, Drive, Sector, Number : word);

ความหมายของพารามิเตอร์แต่ละตัวที่เราจะต้องผ่านค่าให้โพรซีเดอร์ทั้งสองคือ

Segment เซกเมนต์เริ่มต้นของตัวแปรที่ใช้เก็บข้อมูลของเซกเตอร์

Offset ออฟเซตเริ่มต้นของตัวแปรที่ใช้เก็บข้อมูลของเซกเตอร์

Drive หมายเลขไดรฟ์ที่จะอ่านหรือเขียน (0 = ไดรฟ์ปัจจุบัน, 1 = A, 2 = B,...)

Sector หมายเลขเริ่มต้นของลอจิคัลเซกเตอร์ (เซกเตอร์แรกคือเซกเตอร์ 0)

Number จำนวนเซกเตอร์ที่จะอ่านหรือเขียน

ในยูนิต SECTOR__ นี้ยังมีตัวแปรแบบโกลบอลชื่อ ErrorCode อีกหนึ่งตัวไว้สำหรับตรวจสอบข้อผิดพลาด

```
;Procedure ReadSector(Segment,Offset,Drive,Sector,Number: word);
;
;
DATA    SEGMENT
        EXTRN  ErrorCode:byte
DATA    ENDS
;
;
CODE    SEGMENT WORD
ReadSector PROC FAR
        ASSUME CS:CODE, DS:DATA
        PUBLIC ReadSector
        push   bp                  ;save BP
        mov    bp,sp               ;set up stack frame
        push   ds                  ;save DS
        lds    bx,dword ptr[bp+12] ;get buffer address
        mov    al,[bp+10]          ;drive code
        dec    al                  ;adjust for TURBO
        mov    dx,[bp+8]           ;logical sector number
        mov    cx,[bp+6]           ;number sectors to read
        int    25H                 ;read the sector(s)
        mov    bl,0                ;0 = no error
        jnc    L1                  ;test for error
        mov    bl,ah               ;error code to BL
L1:     pop    cx                  ;balance stack
        pop    ds                  ;restore DS
        pop    bp                  ;restore BP
        mov    ErrorCode,bl        ;set error code
        ret    10
ReadSector ENDP
CODE    ENDS
        END
```

รูปที่ 1 การสร้างไฟล์ RDSECTOR.ASM

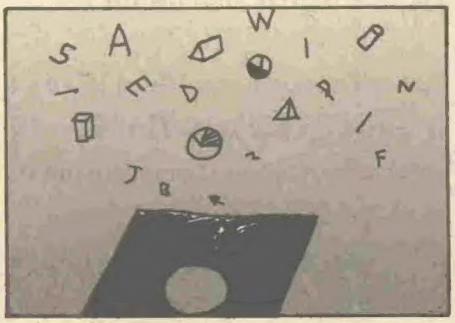
ในการทำงานของโพรซีเดอร์ทั้งสองภายหลังการทำงานของโพรซีเดอร์นี้ค่าใน ErrorCode จะมีความหมายดังนี้

0 ไม่มีข้อผิดพลาด

$03 พยายามเขียนลงดิสค์ที่ป้องกันการเขียน

```
;Procedure WriteSector(Segment,Offset,Drive,Sector,Number: word;);
;
;
DATA    SEGMENT
        EXTRN  ErrorCode:byte
DATA    ENDS
;
;
CODE    SEGMENT WORD
WriteSector PROC FAR
        ASSUME CS:CODE, DS:DATA
        PUBLIC WriteSector
        push   bp                ;save BP
        mov    bp,sp             ;set up stack frame
        push   ds                ;save DS
        lds    bx,dword ptr[bp+12] ;get buffer address
        mov    al,[bp+10]        ;drive code
        dec    al                ;adjust for TURBO
        mov    dx,[bp+8]         ;logical sector number
        mov    cx,[bp+6]         ;number sectors to write
        int    26H               ;write the sector(s)
        mov    bl,0              ;0 = no error
        jnc    L1                ;test for error
        mov    bl,ah             ;error code to BL
L1:     pop    cx                ;balance stack
        pop    ds                ;restore DS
        pop    bp                ;restore BP
        mov    ErrorCode,bl      ;set error code
        ret    10
WriteSector ENDP
CODE    ENDS
        END
```

รูปที่ 2 การสร้างไฟล์ WRSECTOR.ASM



$04 การค้นหาเซกเตอร์ล้มเหลว
$80 ดิสค์ยังไม่พร้อมใช้งาน

รูปแบบในการใช้งานของทั้งสองโพรซีเดอร์จึงเป็นดังนี้

```
{$I-} ReadSector( ) / WriteSecter( ) {$I+}
If ErrorCode = 0 then (Continue)
Else (Check Error)
```

นำ SECTOR_มาใช้งาน

ต่อไปเป็นตัวอย่างการนำยูนิต SECTOR_ ไปสร้างเป็นโปรแกรม View Sector ซึ่งสามารถแสดงข้อมูลภายในเซกเตอร์เป็นอักขระทางแอสกี โดยเลื่อนขึ้นลงทีละเซกเตอร์หรือข้ามไปยังเซกเตอร์ใด ๆ ก็ได้ สามารถลอกข้อมูลของเซกเตอร์หนึ่งไปเขียนลงบนอีกเซกเตอร์หนึ่ง แสดงข้อมูลทีละ 10 ออฟเซตในรูปของตัวเลขฐานสิบ และเขียนข้อมูลของเซกเตอร์ลงไฟล์ได้

เนื่องจากเซกเตอร์จะมีขนาดไม่เกิน 512 ไบต์ และออฟเซตแรกของเซกเตอร์คือออฟเซต 0 ประเภทของตัวแปรที่เราจะใช้เก็บข้อมูลของเซกเตอร์จึงเป็นอะเรย์ของไบต์จาก 0 ถึง 511 โปรแกรมนี้จะเรียงลำดับตามรูปที่ 4, 5,.. ซึ่งพอจะอธิบายคร่าว ๆ ได้ดังนี้

- Procedure CheckError; (รูปที่ 6) ใช้ในการแสดงชนิดของข้อผิดพลาดในการอ่านและเขียนเซกเตอร์

- Procedure CheckFirst; (รูปที่ 7) จะทำการตรวจสอบก่อนว่าไดรฟ์อยู่ในสภาพพร้อมหรือไม่ โดยการทดลองอ่านบูตเซกเตอร์เข้าไปเก็บในตัวแปร TheSector ถ้าผลการอ่านไม่มีข้อผิดพลาดจะทำการตรวจสอบต่อไปว่าดิสค์นั้นมีทั้งหมดกี่เซกเตอร์และขนาดของเซกเตอร์เป็นเท่าใด ด้วยการเรียกใช้อินเตอร์รัพต์ 21H ฟังก์ชัน 36H

รูปที่ 3 ไฟล์ SECTOR.PAS

```
Unit SECTOR_;
Interface
Type
  str11 = string[11];
Var
  ErrorCode : byte;

  Procedure ReadSector(Segment,Offset,Drive,Sector,Number: word);
  Procedure WriteSector(Segment,Offset,Drive,Sector,Number: word);

Implementation
  Procedure ReadSector(Segment,Offset,Drive,Sector,Number: word);external;
  {$L RDSECTOR.OBJ}
  Procedure WriteSector(Segment,Offset,Drive,Sector,Number: word);external;
  {$L WRSECTOR.OBJ}

end.
```

รูปที่ 4 ตัวอย่างการนำยูนิต SECTOR_สร้างเป็นโปรแกรม View Sector

```
PROGRAM  ViewSector;
USES  Crt, Dos, Sector_;
TYPE  SecType                    = Array [0..511] of Byte;
VAR   TheSector                  : SecType;
      Drive, LastDrive           : Byte;
      DriveCh, Kar               : Char;
      CanQuit, First, OfToSector : Boolean;
      W, Sector, ToSector        : Word;
      OldSector, LastSector      : Word;
      R                          : Registers;
      BytePerSector              : Word;
```

รูปที่ 5

```
PROCEDURE  ReadKar;
BEGIN
  Kar:= ReadKey; If Kar = #0 then Kar:= ReadKey;
END;
```

ของดอส ค่าที่เก็บในตัวแปรชื่อ LastSector จะเท่ากับจำนวนเซกเตอร์ -1 เนื่องจากหมายเลขของลอจิคัลเซกเตอร์เริ่มจาก 0 ส่วนค่าที่เก็บในตัวแปรชื่อ BytePerSector จะเท่ากับขนาดของเซกเตอร์ -1 เช่นกัน (เนื่องจากหมายเลขออฟเซตจะเริ่มจาก 0)

- Procedure AskQuit; (รูปที่ 8) โพรซีเดอร์นี้จะถูกเรียกใช้เมื่อเรากดปุ่ม Esc เพื่อถามผู้ใช้โปรแกรมว่าจะออกจากโปรแกรมหรือไม่

- Procedure DispSector; (รูปที่ 9) นับเป็นหัวใจของโปรแกรมนี้เลยทีเดียว เพราะภายในจะมีโพรซีเดอร์ย่อย BackWindow เพื่อกลับเข้าสู่หน้าต่างการแสดงข้อมูลในเซกเตอร์ การทำงานของโพรซีเดอร์นี้จะเริ่มจากการแสดงผลข้อมูลในเซกเตอร์ 0 (บูตเซกเตอร์) ก่อน แล้วรอรับการกดปุ่มเพื่อเลือกการทำงานต่อไป ได้แก่

Esc {#27} ต้องการออกจากโปรแกรม

F1 {#59} แสดงข้อมูลในเซกเตอร์ 10 ออฟเซตเริ่มจากออฟเซตที่ต้องการ

Up Arrow {#72} เลื่อนขึ้น 1 เซกเตอร์

Dn Arrow {#80} เลื่อนลง 1 เซกเตอร์

C นำข้อมูลของเซกเตอร์นี้ไปเขียนลงในเซกเตอร์อื่น ซึ่งอาจเขียนทับเซกเตอร์เดิมหรือเขียนลงดิสค์แผ่นอื่นก็ได้

F นำข้อมูลของเซกเตอร์นี้เขียนลงไฟล์ SECTOR.OUT ในไดเรคทอรีปัจจุบันของดิสค์ A

G ข้ามไปยังเซกเตอร์ที่ต้องการ แต่ถ้าเราป้อนค่าเซกเตอร์ที่ไม่มีอยู่จริง จะทำให้โปรแกรมกลับไปเริ่มต้นใหม่

```
PROCEDURE  CheckError;
BEGIN
  Window(1,15,80,25); ClrScr;
  Case ErrorCode of
    4 : begin
          Write(#7,'Sector ');
          If OfToSector then Write(ToSector)
          Else Write(Sector);
          WriteLn(' does not exist !');
          Write('Press any key...');
          ReadKar;
          First:= true;
        end;
  128 : begin
          WriteLn(#7,'Drive ',DriveCh,' is not ready !');
          Write('Press any key when ready');
          ReadKar;
          First:= true;
        end;
  Else  begin
          WriteLn(#7,'Sector read/write error !');
          Write('Press any key...');
          First:= true;
          ReadKar;
        end;
  End; { Case }
  ClrScr
END;
```

รูปที่ 6 โพรซีเดอร์สำหรับแสดงชนิดของข้อผิดพลาดในการอ่านและเขียนเซกเตอร์

(เราอาจแก้ไขในส่วนนี้ให้เปรียบเทียบค่าที่รับเข้ามากับค่าของ LastSector ก็ได้)

- {Main} (รูปที่ 10) ในส่วนนี้จะเป็นการกำหนดค่าเริ่มต้นให้กับตัวแปรต่าง ๆ ตรวจสอบจำนวนดิสค์ที่มีอยู่ โดยใช้อินเตอร์รัพต์ 11H ของดอสเสร็จแล้วเข้าสู่การทำงานของโปรแกรม

ใช้งาน View Sector

การแสดงผลของโปรแกรมนี้จะเป็นดังรูปที่ 11 ซึ่งเราสามารถจะนำไปประยุกต์ให้เกิดประโยชน์ได้หลายทาง คือ

- ตรวจสอบไวรัสในบูตเซกเตอร์ เช่น C Brain, Stoned ฯลฯ

- ป้องกันบูตเซกเตอร์ไม่ให้ติดไวรัสบางตัว เช่น C Brain โดยการเข้าไปแก้ไขบางส่วนของบูตเซกเตอร์ คือ เปลี่ยนออฟเซตที่ 4 และ 5 ให้เป็น $34 และ $12 ตามลำดับ แล้วเขียนทับลงไปใหม่

- แก้ไขบูตเซกเตอร์ที่ติดไวรัส ซึ่งอาจทำได้โดยก๊อปปี้บูตเซกเตอร์จากแผ่นที่มีฟอร์แมตเหมือนกันและไม่ติดไวรัสมาเขียนทับลงไป หรือก๊อปปี้จากเซกเตอร์ที่ไวรัสย้ายบูตเซกเตอร์ไปเก็บ

รูปที่ 7 ตรวจสอบความพร้อมของไดรฟ์

```
PROCEDURE  CheckFirst;
BEGIN
  First:= False;
  {$I-}
  ReadSector(Seg(TheSector),Ofs(TheSector),Drive,0,1); {$I+}
  If ErrorCode = 0 Then
    begin
```

รูปที่ 7 (ต่อ)

```
    Window(1,3,80,13);
    R.dl:= Drive;
    R.ah:= $36;
    MSDos(R);
    LastSector:= ((((R.ax*R.dx) div 40) +1) *40) -1;
    BytePerSector:= R.cx-1
  end
 Else CheckError;
END;
```

```
PROCEDURE  AskQuit;
BEGIN
  Write('Quit ? <Y/N>');
  ReadKar;
  If UpCase(Kar) = 'Y' Then CanQuit:= True
  Else First:= True;
END;
```

รูปที่ 8 โพรซีเดอร์นี้จะใช้เมื่อต้องการออกจากโปรแกรม

รูปที่ 9 การทำงานของโพรซีเดอร์นี้จะเริ่มจากการแสดงผลข้อมูลในเซกเตอร์ 0 เพื่อเลือกการทำงานต่อไป

```
PROCEDURE  DispSector;
VAR  Kar2   : Char;
     OfsAsk : Word;
     Fi     : File of SecType;
Procedure  BackWindow;
Begin
  ClrScr;
  Window(1,3,80,13);
  GotoXY(43,11)
End;
BEGIN
  Kar2:= ' ';
  Repeat
    If First then CheckFirst
    Else ErrorCode:= 0;
    If ErrorCode = 0 Then
    begin
      {$I-}
      ReadSector(Seg(TheSector),Ofs(TheSector),Drive,Sector,1); {$I+}
      If ErrorCode = 0 Then
      begin
        GotoXY(1,1); ClrEOL; WriteLn('Sector ',Sector);
        Write('
);
        {Write 80 Columns}
        For W:= 0 to BytePerSector Do
```

รูปที่ 9 (ต่อ)

```
          If TheSector[W] in [7,8,10,13,27] then Write('.')
          Else Write(Chr(TheSector[W]));
        WriteLn;
        Write('═══════════════════════════════════════════════════════════════════════════════');
);
        Write('Hot key : ',#24,' ',#25,' ','Esc F1 C(opy) G(oto) F(ile) ');
      end
      Else CheckError;
    end
  Until ErrorCode <> 128;
  If ErrorCode = 0 Then
  Repeat
    OldSector:= Sector;
    repeat
      ReadKar
    until UpCase(Kar) in [#27,#59,#72,#80,'C','F','G'];
    Kar2:= UpCase(Kar);
    Case Kar2 of
      #27 : begin
              Window(1,1,80,25); GotoXY(1,14); AskQuit
            end;
      #59 : Begin
              Window(1,15,80,25);
              Write('10 Bytes from Offset ');
              ReadLn(OfsAsk);
              For W:= OfsAsk to OfsAsk+9 Do Write(TheSector[W],' ');
              WriteLn; Write('Press any key to continue...');
              ReadKar;
              BackWindow;
            End;
      #72 : If Sector > 0 then Dec(Sector)
            Else Write(#7);
      #80 : If Sector < LastSector then Inc(Sector)
            Else Write(#7);
      'C' : Begin
              Window(1,15,80,25);
              Write('Copy this sector to sector ');
              ReadLn(ToSector); OfToSector:= True;
              {$I-} WriteSector(Seg(TheSector),Ofs(TheSector),
                   Drive,ToSector,1); {$I+}
```

รูปที่ 9 (ต่อ)

```
                If ErrorCode <> 0 then CheckError;
                BackWindow
              End;
        'F' : Begin
                Window(1,15,80,25);
                WriteLn('Write this sector to file ''A:SECTOR.OUT.''');
                Write('Press any key when ready...');
                ReadKar;
                Assign(Fi,'A:SECTOR.OUT');
                {$I-} Rewrite(Fi); {$I+}
                If IOresult = 0 then Write(Fi,TheSector);
                BackWindow
              End;
        'G' : Begin
                Window(1,15,80,25);
                Write('Goto sector '); ReadLn(Sector);
                BackWindow
              End
      End
    Until (Sector <> OldSector) or (Kar2 = #27) or OfToSector;
    OfToSector:= False;
  END;
```

ไว้มาเขียนทับลงไปก็ได้ ทั้งนี้ในการใช้งานโปรแกรมนี้จะต้องบูตเครื่องโดยใช้แผ่นที่ไม่มีไวรัส มิฉะนั้นโปรแกรมอาจถูกไวรัสบางตัวล่อหลอก เช่น C Brain หลอกว่าบูตเซกเตอร์ยังปกติอยู่ได้

- ช่วยศึกษาโครงสร้างบูตเซกเตอร์ ตารางการจัดสรรแฟ้ม (FAT) และตารางควบคุมแฟ้ม (directory)

- ใช้กู้ไฟล์ในดิสค์ที่เกิด cross-linked files โดยใช้คำสั่ง CHKDSK /F ของดอสก่อน หากได้ข้อมูลที่ต้องการมาไม่ครบให้ลองหาข้อมูลส่วนนั้นเองแล้ว File ลงในดิสค์ (เราอาจปรับปรุง

รูปที่ 10 กำหนดค่าเริ่มต้นให้กับตัวแปรเพื่อตรวจสอบจำนวนดิสค์ที่มีอยู่

```
BEGIN { Main }
  CanQuit:= False; First:= True;
  OfToSector:= False;
  Intr($11,R);
  LastDrive:= (R.al shr 6 + 1) + 64;
  Repeat
    If First then
    begin
      Sector:= 0;
      Window(1,1,80,25); ClrScr;
      WriteLn('▒▒  VIEW_SECTOR  ▒▒ ',
              'by : Pradit Pinyopasakul Intania 73 CU.');
```

รูปที่ 10 (ต่อ)

```
    Write('Drive ');
    repeat
      ReadKar;
    until ((Ord(Upcase(Kar)) > 64) and (Ord(UpCase(Kar)) <= LastDrive))
          or (Kar = #27);
    If Kar = #27 Then
      Begin
        GotoXY(1,4);
        AskQuit;
        if CanQuit then
          begin ClrScr; Halt end
        else DelLine
      End
    Else
      Begin
        DriveCh:= UpCase(Kar);
        WriteLn(DriveCh);
        Drive:= Ord(DriveCh)-64;
        Window(1,3,80,25);
      End;
end;
DispSector;
  Until CanQuit;
  Window(1,1,80,25); ClrScr
END.
```

โปรแกรมต่อให้สามารถกำหนดชื่อไฟล์ที่จะเขียนลงดิสค์ได้ไม่ยาก) วิธีนี้ต้องใช้เวลาสักหน่อย แต่ก็คุ้มค่ามิใช่หรือ ผู้เขียนเองยังเคยโดนไฟล์โปรแกรมขนาดกว่า 1 พันบรรทัดเล่นงาน โชคดีที่โดนเพียง 70 บรรทัดสุดท้าย (ประมาณ 3 เซกเตอร์) จึงสามารถกู้คืนมาได้จนครบ

ทิ้งท้าย

ต่อไปนี้การอ่านและเขียนกับเซกเตอร์โดยตรงคงไม่เป็นปัญหาสำหรับผู้ใช้เทอร์โบปาสคาลอีกต่อไปแล้ว ขอให้สนุกกับการนำยูนิต SECTOR__ ไปใช้เขียนโปรแกรมต่าง ๆ ซึ่งอาจจะเป็นโปรแกรมวัคซีนของเราเองก็ได้ เราจะได้ช่วยกันปราบเจ้าไวรัสที่ทำให้เราใช้งานคอมพิวเตอร์อย่างไม่เป็นสุขให้หมดสิ้นเสียที

รูปที่ 11 การแสดงผลของโปรแกรม View Sector

```
VIEW_SECTOR  by : Pradit Pinyopasakul Intania 73 CU.
Drive B
Sector 27
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```